DP-304828

GENERATION OF MULTIPLE INDEPENDENT HIGH RESOLUTION PULSE WIDTH MODULATIONS

Melissa M. Klemish
Robert B. Gager
Lamar H. McLouth

DP-304828

GENERATION OF MULTIPLE INDEPENDENT HIGH RESOLUTION PULSE WIDTH MODULATIONS

Technical Field

[0001] The present invention relates to pulse width modulated (PWM) signals and more particularly to high-resolution PWM signals.

Background of the Invention

[0002] PWM signals are used in a variety of applications. Some examples of typical applications include controlling motor speed, controlling lighting intensity, and controlling the movement of a gage pointer to name just a few.

[0003] Pulse width modulation is the changing of a signal's duty cycle within a fixed period. The resolution of a PWM signal is defined by the number of different duty cycles in the signal and the number of steps at which the signal transitions within a period. For example, a PWM signal having a duty cycle of 50% is on half of the time and off half of the time while a PWM signal having duty cycle of 75% is on three-quarters of the time and off one-quarter of the time. The number of steps at which the signal transitions within a period is its resolution, 2^n. A PWM signal having 8-bit resolution can have 256, or 2^8, different duty cycles, while a 10 bit resolution signal has 1024, or 2^{10}, different duty cycle values. High-resolution is typically a resolution greater than 8-bit.

[0004] PWM signals are generated using hardware such as a microprocessor integrated peripheral. The peripheral device is limited in the number of PWM signals that it can generate, and is typically limited to four or fewer signals. Hardware is available that can generate more than four signals, but becomes very expensive and therefore, is not always an option for many

applications. Many applications for PWM signals require more than four signals, an example being as many as 15 signals that are required for an application such as an automotive instrument cluster. However, a more expensive processor is not necessarily a cost-effective solution for many applications.

[0005] There is a need for a simple, inexpensive method of generating high-resolution PWM signals without the need for expensive additional hardware.

Summary of the Invention

[0006] It is an object of the present invention to generate multiple independent high-resolution pulse width modulations. It is another object of the present invention to provide software that is compatible with standard microprocessors in order to generate high-resolution PWM signals. It is yet another object of the present invention to provide a software program that manipulates microprocessor hardware to generate multiple high-resolution PWM signals.

[0007] The present invention introduces software interrupts at predetermined intervals to generate multiple high-resolution PWM signals from a conventional microprocessor. The software determines the purpose of the interrupt in an interrupt subroutine. An interrupt occurs when a compare register equals the timer. Another interrupt occurs when the timer overflows. The interrupts are the key to generating the multiple high-resolution PWM signals.

[0008] The software of the present invention creates a PWM generation table that has a unique entry for each unique duty cycle and the port values at the time that a unique duty cycle is compared with the timer. At the end of the table, there is a value called an invalid duty cycle value, which

allows the timer to overflow. The PWM generation table is then used to generate the interrupts and the PWM signals.

[0009] The present invention provides a low cost method of generating high-resolution PWM signals and can be incorporated with any microprocessor having a capture and compare module. According to the present invention, the number of PWM signals is not limited by the hardware microprocessor as in the prior art where each PWM required one more set of PWM hardware. The number of PWM signals in the present invention is limited only by the number of general input and output ports available on the processor.

[0010] Other objects and features of the present invention will become apparent when viewed in light of the detailed description of the preferred embodiment when taken in conjunction with the attached drawings and appended claims.

Brief Description of the Drawings

[0011] In order that the present invention may be well understood, there will now be described some embodiments thereof, given by way of example, with reference to the accompanying drawings, in which:

[0012] FIGURE 1 is a prior art example of a typical capture and compare module of a microprocessor used to generate a single PWM signal;

[0013] FIGURE 2 is a block diagram of the processor components utilized in the present invention and their relationship to each other;

[0014] FIGURE 3 is a flow chart of the software algorithm of the present invention;

[0015] FIGURE 4 is a flow chart of a sort routine for building a table of duty cycle values according to the present invention; and

[0016] FIGURE 5 is a flow chart of an interrupt subroutine used in sorting the duty cycle value table according to the present invention.

Description of the Preferred Embodiment

[0017] Figure 1 is a prior art example of a typical capture and compare module 10 used in the generation of a single PWM signal. To generate a single PWM signal, a duty cycle value 12 is compared, by way of a comparator 14, to a continuous timer 16. In the present example, the timer 14 is a 16-bit timer that counts from 0 to 65535.

[0018] When the duty cycle value 12 is equal to the value of the count in the timer 16, output logic 18 sets a predefined hardware pin 20 high. When the timer overflows from 65535 to zero, the output logic 18 sets pin 20 low again. In this regard, changing the duty cycle value 12 in a duty cycle register, (not shown), changes the duty cycle of the PWM signal. An example signal 22 is shown in Figure 1. Changing the point at which the timer starts counting changes the period. For example, changing the beginning of the count from 0, to 300 would shorten the period, and increase the frequency of the PWM signal.

[0019] The present invention uses a software program to manipulate the hardware described above in order to generate multiple high-resolution PWM signals. For example purposes, the invention will be described as it generates fifteen high-resolution PWM signals. However, one skilled in the art is capable of generating more or fewer signals without departing from the scope of the present invention.

[0020] Referring now to Figure 2, the present invention is used in conjunction with an 8-bit processor 30 that manipulates a 16-bit timer 32, a Universal Asynchronous Receiver/Transmitter (UART) communication bus 34 and the capture and compare module 10. According to the present invention,

the capture and compare module 10 generates a software interrupt when its compare register equals the timer value. While a 16HC73B PIC 8-bit processor and a 16-bit timer are shown and described herein, it should be noted that any processor with a communication bus, a capture/compare module, or a PWM module, and a timer can be used in accordance with the present invention.

[0021] The processor 30 receives duty cycle values from any one of many sources such as another microcontroller, a personal computer, or any other device capable of transmitting data by way of the UART communication bus 34.

[0022] A block diagram of the software 100 of the present invention is shown in Figure 3. The I/O ports, UART communication and interrupts are all initialized 102. The program receives 104 data and then places 106 the data in a table that represents the duty cycle values. The program then checks 108 to see if all data has been received. The program will continue to receive data 104 and fill 106 the table until all data is present. Once the table is full, the software calls 110 a sort routine where a PWM generation table is created. A pointer is set 111 to the beginning of the table and the program waits for the receipt 104 of new data.

[0023] When an interrupt is generated 112, the software decides 114 what type of interrupt occurred and reacts accordingly. An interrupt is generated from two different sources according to the present invention. A capture and compare interrupt and a timer overflow interrupt. For a capture and compare interrupt generated by a match between the timer and the duty cycle register, the program generates the PWM signals 116 as defined by the PWM generation table. In the event the interrupt is a result of the timer overflow 118, the program resets the ports and returns to the beginning of the PWM generation table 120. Finally, the exit routine is exited 122.

[0024] Figure 4 is a block diagram of the sort routine 110. The data that has been received is placed in a duty cycle table 212 and sorted along with data that is stored in two other tables embedded in the software. A table 214 contains corresponding ports for each duty cycle and a table 216 contains corresponding bit masks for the specific port pin, also embedded in the software. The duty cycle table 212 is immediately over written upon receipt of new data. The port table 214 and the bit-mask table 216 have constant values that depend on the specific application.

[0025] The software sorts 218 the table 212 of duty cycle values from lowest to highest. The entries in the second 214 and third 216 tables are moved with their corresponding entries from the duty cycle table 212 in order to build a PWM generation table 220 that is used in the interrupt routine. The PWM generation table 220 contains the duty cycle values, the port, and the time that the duty cycle values will be at the ports.

[0026] The sort routine 110 will combine like values and ignore invalid values when building the PWM generation table 220. For example, if all the ports have the same duty cycle, there will be only one entry in the PWM generation table 220. There is a separate entry for each duty cycle. At the end of the PWM generation table, there is an invalid duty cycle value. The invalid duty cycle value is a value that can never be equal to the timer. The presence of this value in the table allows the timer to overflow.

[0027] Referring again to Figure 3, in the present invention, all of the receiving 104, 106, 108 and sorting 110 is done in the processor's background memory. The PWM signals are generated during an interrupt routine, which is entered upon generation 112 of an interrupt. During the interrupt routine, for a capture and compare interrupt 116, the port pins are set high and low to generate the PWM signals. The interrupt routine is entered only when the duty cycle register matches the timer, or when the timer overflows. Because

there is only one interrupt, the processor must decide why it received that interrupt.

[0028] Referring to Figure 5, there is shown a detailed flow diagram of the interrupt routine. When an interrupt is generated 112, the program determines if the interrupt is due to a match between the timer value and the duty cycle value. If so, this is considered a CAPCOM interrupt 114 and the next step is to write values 116 to a compare register and the pin ports directly from the PWM generation table.

[0029] Should the program determine the interrupt is a result of the timer overflowing 118, then the program 120 writes zeros to the PWM port and returns to the beginning of the PWM generation table. Ultimately, the interrupt routine is exited 122.

[0030] Referring again to Figure 4, during initialization, the processor configured and started the capture/compare module and places the first two values of the PWM generation table into the duty cycle register, CCPR1H and CCPR1L. Referring back to Figure 3, when the timer equals the values in the CCPR1H and CCPR1L registers, an interrupt will occur and the interrupt routine 112 is initiated. The interrupt sets the appropriate ports high. This being previously determined in the sort routine. The interrupt routine 112 does not need to determine which pins are set. It merely reads from the PWM generation table and places values to other registers, i.e. PORTA, PORTB, PORTC, CCPR1H, and CCPR1L to set up the next time it should interrupt.

[0031] Once the interrupt routine is exited, it will not be entered again until the timer equals the new values in the CCPR1H and CCPR1L registers. This will continue until the invalid values that were placed at the end of the PWM generation table are loaded into CCPR1H and CCPR1L. These values are invalid because they are values to which the timer will never equal.

Therefore, an interrupt will not be generated and the next interrupt will occur when the timer overflows from 65535 to zero.

[0032] At this point in time, all of the ports are set to low and a pointer is returned to the beginning of the PWM generation table. The first two values placed in the PWM generation table are placed in the duty cycle registers, CCPR1H and CCPR1L, and the cycle begins again.

[0033] According to the present invention, and assuming a 20 MHz external clock, the resolution obtained for a 240 Hz period is approximately 9-bits, for a 120 Hz period, it is approximately 10-bits and for a 75 Hz period, the resolution is over 10-bits. Multiple PWM signals having such high resolutions are possible because of the fact that the processing and table generation is done in a background task.

[0034] While particular embodiments of the invention have been shown and described, numerous variations and alternate embodiments will occur to those skilled in the art. Accordingly, it is intended that the invention be limited only in terms of the appended claims.